Windows Programming

Visual C++ MFC Programming

Lecture 03

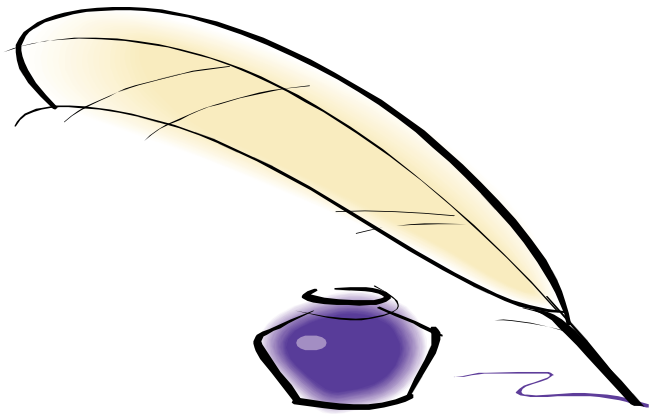
김예진
Dept. of Game Software

Notices

- 03/07: 502 → 501 등록 이동
- 03/21: HW 1 (Due: 03/28)

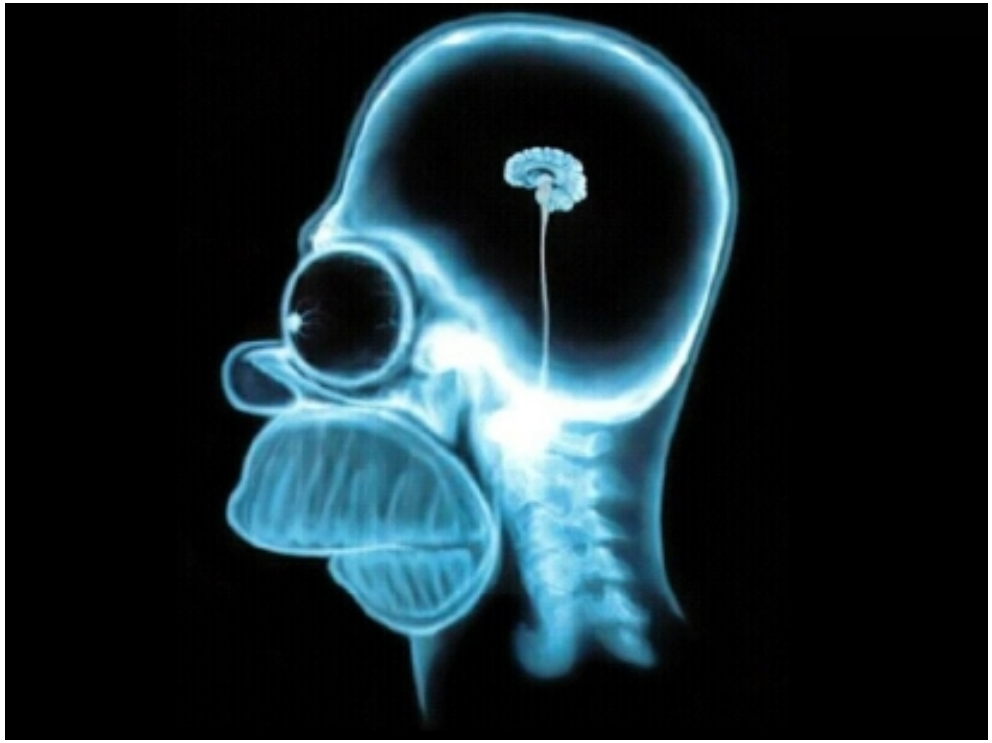
Plan

- MFC Program 구조
  - MFC 기초 Class
- MFC 화면 출력
  - GDI와 DC
  - CPaintDC 사용
  - CClientDC 사용
  - Windows CDC Class



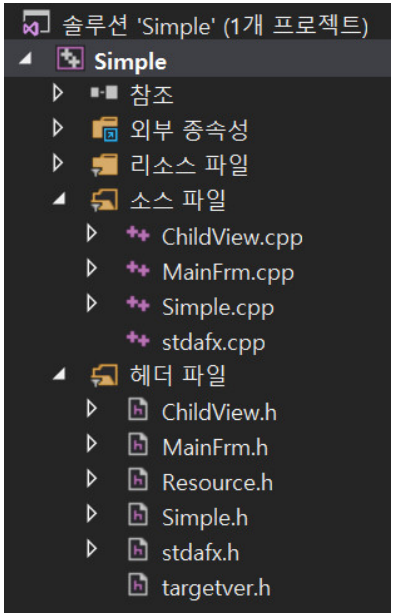
Look into the codes

classes



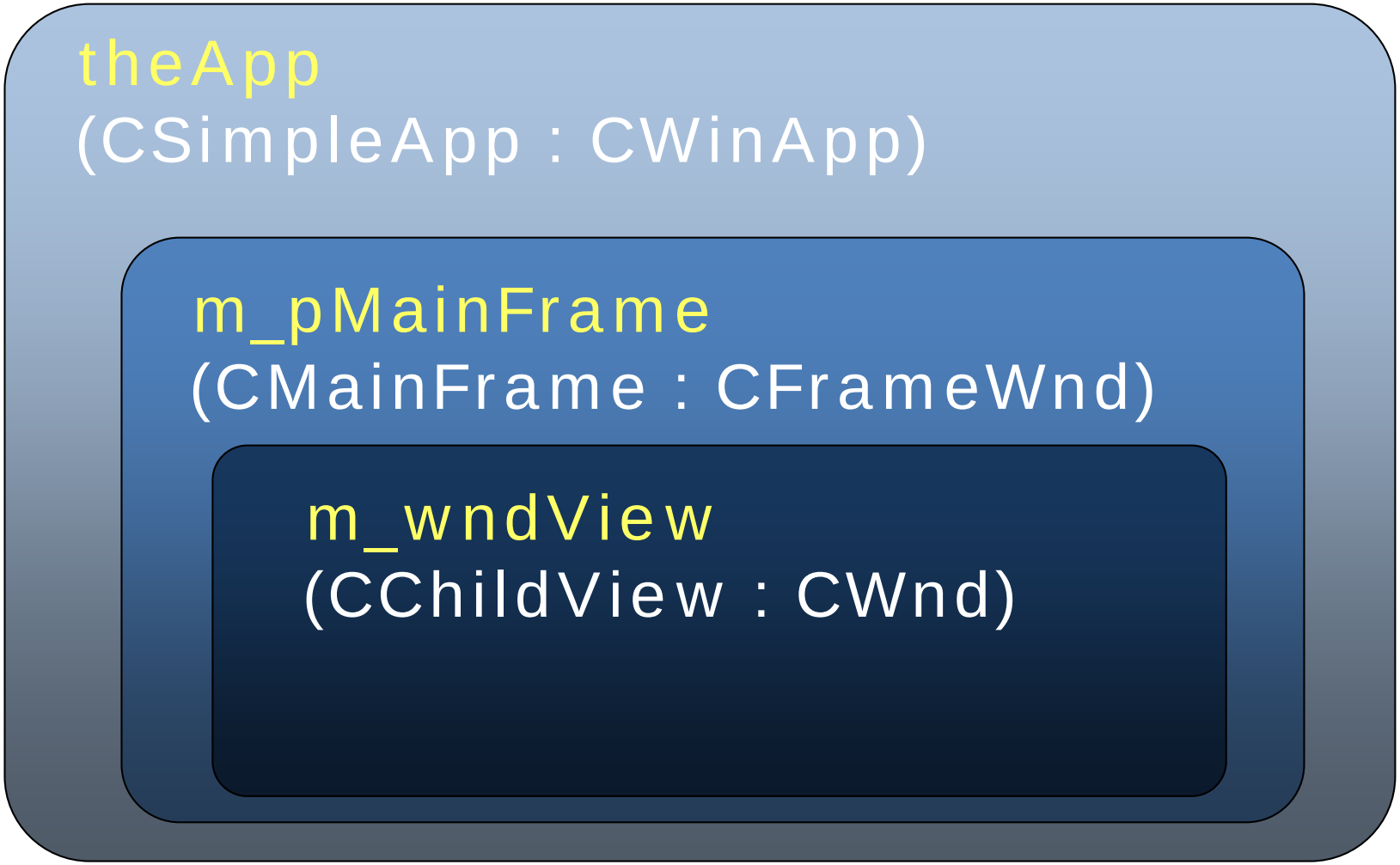
MFC Frame 구조

- AppWizard: MFC 응용 프로그램 생성
  - 새 프로젝트 → Visual C++ → MFC 응용 프로그램 (이름: Simple)
  - MFC 응용 프로그램
    1. 응용 프로그램 종류: 단일 문서, 문서/뷰 아키텍처 지원 (X)
    2. 문서 템플릿 속성: 변경 없음
    3. 사용자 인터페이스 기능: Classic menu options: <없음>
    4. 고급 기능: 인쇄 및 인쇄 미리 보기 (X), ActiveX 컨트롤 (X)
    5. 생성된 클래스: CSimpleApp



MFC Frame 구조

- MFC program 내부 구조

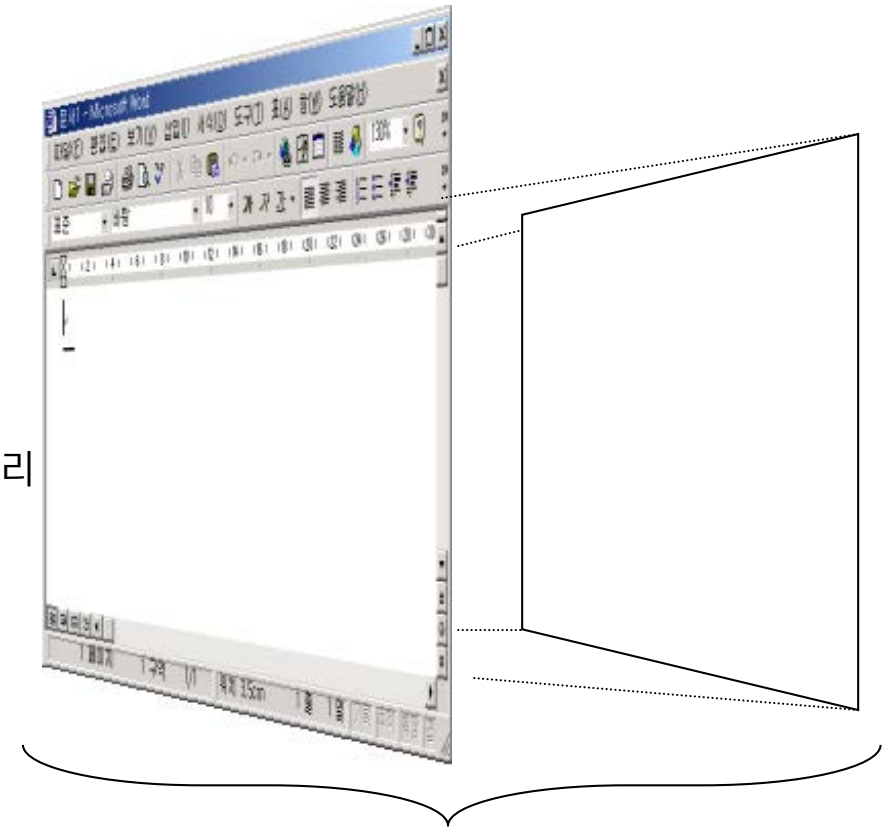


MFC Frame 구조

// MainFrm.h

CMainFrame : CFrameWnd

- Window의 frame(틀)을 관리



// ChildView.h

CChildView : CWnd

- Data를 보여주는 Window

// Simple.h

CSimpleApp : CWinApp

- 위의 두 object를 묶어 주고, program을 구동 시킴 (눈에 안보임)
- Message loop를 돌림

MFC Frame 구조

- Program 실행 순서
  - 미리 약속된 member 함수들이 필요한 순간에 순서대로 호출됨
  - 함수들의 재정의를 통해 원하는 작업을 수행함

```
// Simple.h
Class CSimpleApp : public CWinApp
{
    ...
}

extern CSimpleApp   theApp;
```

```
WinMain()                                   // MFC 내부에 숨겨져 있음
{

        theApp.InitInstance();              // 초기화

        theApp.Run();                       // Message loop

        theApp.ExitInstance();              // 종료

}
```

MFC Frame 구조

- Main program class: CSimpleApp (1/3)

```
// Simple.h
class CSimpleApp : public CWinApp
{
      public:
          CSimpleApp();                          // Class 생성자

          virtual BOOL InitInstance();           // 초기화
          virtual BOOL ExitInstance();           // 종료

          afx_msg void OnAppAbout();             // About message handler

          DECLARE_MESSAGE_MAP()                  // 1개이상의 message handler
};
```

```
// Simple.cpp
#include "stdafx.h"                 // MFC 기본 header file (항상 제일 먼저 나와야함)
#include "afxwinappex.h"            // MFC 관련 header file (afx로 시작함)
...

BEGIN_MESSAGE_MAP(CSimpleApp, CWinApp)      // 지정한 ID에 대한 실행되는 message handler 함수
    ON_COMMAND(ID_APP_ABOUT, &CSimpleApp::OnAppAbout)
END_MESSAGE_MAP()

CSimpleApp::CSimpleApp()
{
}

CSimpleApp theApp;                  // 응용프로그램 자신에 해당하는 전역객체
```

MFC Frame 구조

- Main program class: CSimpleApp (2/3)

```
// Simple.cpp
BOOL CSimpleApp::InitInstance()
{
    SetRegistryKey(_T("로컬 응용 프로그램 마법사에서 생성된 응용 프로그램"));

    CMainFrame* pFrame = new CMainFrame;

    m_pMainWnd = pFrame;

    pFrame->LoadFrame(IDR_MAINFRAME,        // F1키를 누르면 해당 Window API에 대한 설명이 나옴
        WS_OVERLAPPEDWINDOW | FWS_ADDTOTITLE, NULL,
        NULL);

    pFrame->ShowWindow(SW_SHOW);
    pFrame->UpdateWindow();

    return TRUE;
}
```

MFC Frame 구조

- Main program class: CSimpleApp (3/3)

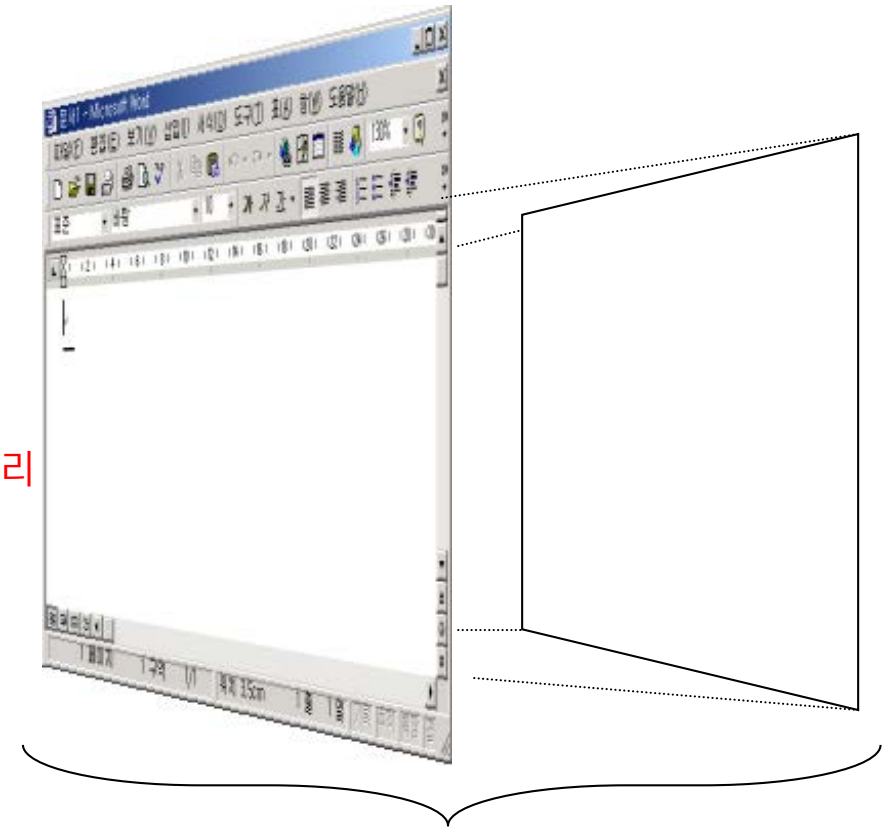
```
// Simple.cpp
BEGIN_MESSAGE_MAP(CAboutDlg, CDialogEx)
END_MESSAGE_MAP()

// 대화 상자를 실행하기 위한 응용 프로그램 명령입니다.
void CSimpleApp::OnAppAbout()
{
CAboutDlg aboutDlg;
aboutDlg.DoModal();
}

// CSimpleApp 메시지 처리기
```

MFC Frame 구조

// MainFrm.h

CMainFrame : CFrameWnd

- Window의 frame(틀)을 관리



// ChildView.h

CChildView : CWnd

- Data를 보여주는 Window

// Simple.h

CSimpleApp : CWinApp

- 위의 두 object를 묶어 주고, program을 구동 시킴 (눈에 안보임)
- Message loop를 돌림

MFC Frame 구조

- Main frame class: CMainFrame (1/2)
  - On으로 시작하는 함수: Message Handler
  - 미리 정의된(약속된) 함수들의 재정의

```
// MainFrm.h
class CMainFrame : public CFrameWnd
{
public:
    CMainFrame();

protected:
    DECLARE_DYNAMIC(CMainFrame);

public:
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs); // Program 실행 초기화 정의
    virtual BOOL OnCmdMsg(UINT nID, int nCode, void* pExtra,
    AFX_CMDHANDLERINFO* pHandlerInfo);
    virtual ~CMainFrame();
    CChildView m_wndView;

protected:
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct); // Window 생성 handler
    afx_msg void OnSetFocus(CWnd *pOldWnd);
    DECLARE_MESSAGE_MAP()
};
```

MFC Frame 구조

- Main frame class: CMainFrame (2/2)
  - On으로 시작하는 함수: Message Handler
  - 미리 정의된(약속된) 함수들의 재정의

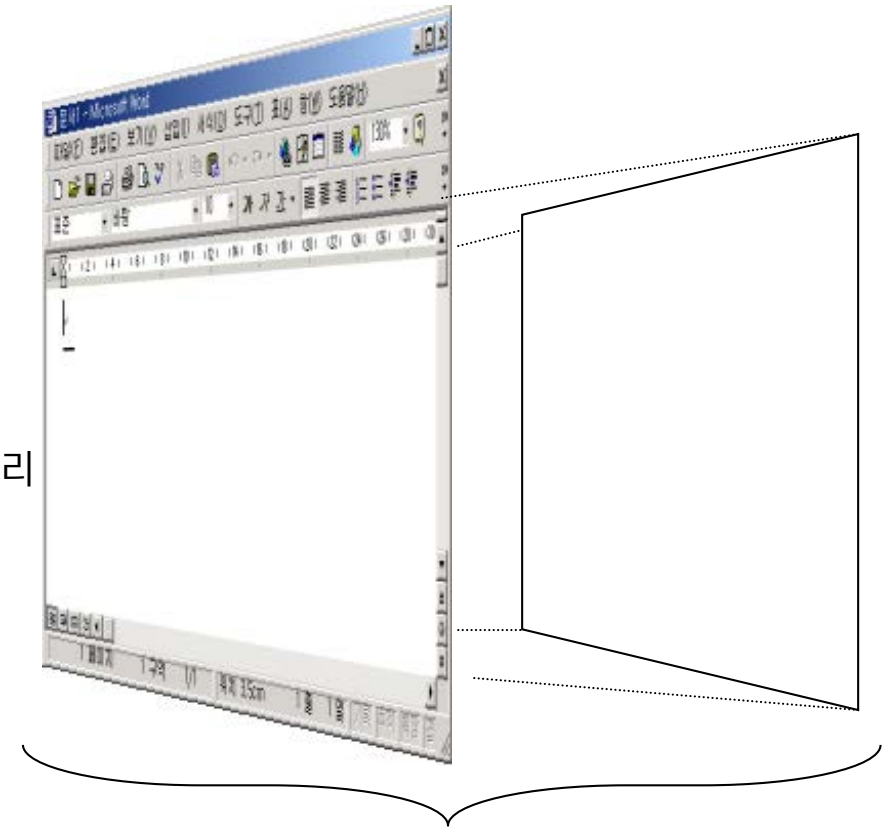
```
// MainFrm.cpp
IMPLEMENT_DYNAMIC(CMainFrame, CFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
    ON_WM_CREATE()
    ON_WM_SETFOCUS()
END_MESSAGE_MAP()

CMainFrame::CMainFrame() { }

CMainFrame::~CMainFrame() { }

int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
       …
       if (!m_wndView.Create(NULL, NULL, AFX_WS_DEFAULT_VIEW,
             CRect(0, 0, 0, 0), this, AFX_IDW_PANE_FIRST, NULL))
         {
             …
         }
             …
       return 0;
}
```

MFC Frame 구조

// MainFrm.h

CMainFrame : CFrameWnd

- Window의 frame(틀)을 관리



// ChildView.h

CChildView : CWnd

- Data를 보여주는 Window

// Simple.h

CSimpleApp : CWinApp

- 위의 두 object를 묶어 주고, program을 구동 시킴 (눈에 안보임)
- Message loop를 돌림

MFC Frame 구조

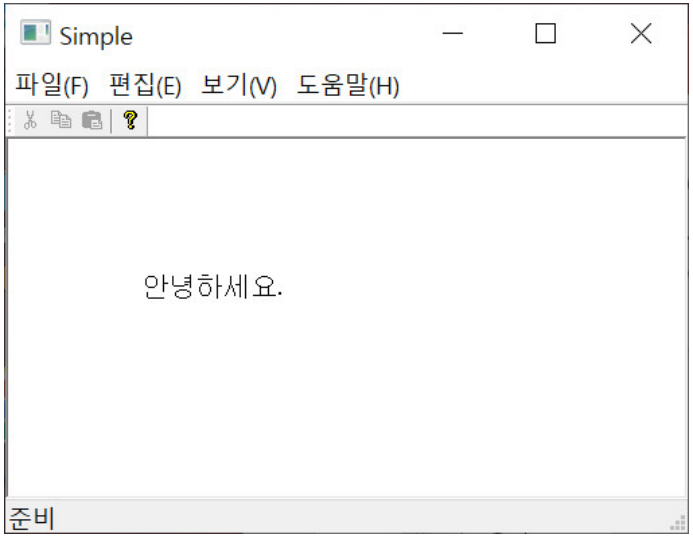
- View class: CChildView (1/2)
  - On으로 시작하는 함수: Message Handler
  - 미리 정의된(약속된) 함수들의 재정의

```
// ChildView.h
class CChildView : public CWnd
{
public:
    CChildView();

protected:
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);

public:
    virtual ~CChildView();

protected:
    afx_msg void OnPaint(); // Window client에 그림을 그리는 handler
    DECLARE_MESSAGE_MAP()
};
```

MFC Frame 구조

- View class: CChildView (2/2)
  - On으로 시작하는 함수: Message Handler
  - 미리 정의된(약속된) 함수들의 재정의

```
// ChildView.cpp
CChildView::CChildView()
{
}

CChildView::~CChildView()
{
}

BEGIN_MESSAGE_MAP(CChildView,CWnd )
    ON_WM_PAINT()
END_MESSAGE_MAP()

void CChildView::OnPaint()
{
     CPaintDC dc(this); // 그리기를 위한 디바이스 컨텍스트입니다.

     // TODO: 여기에 메시지 처리기 코드를 추가합니다.
     dc.TextOut(100, 100, _T("안녕하세요."));
     // 그리기 메시지에 대해서는 CWnd::OnPaint()를 호출하지 마십시오.
}
```



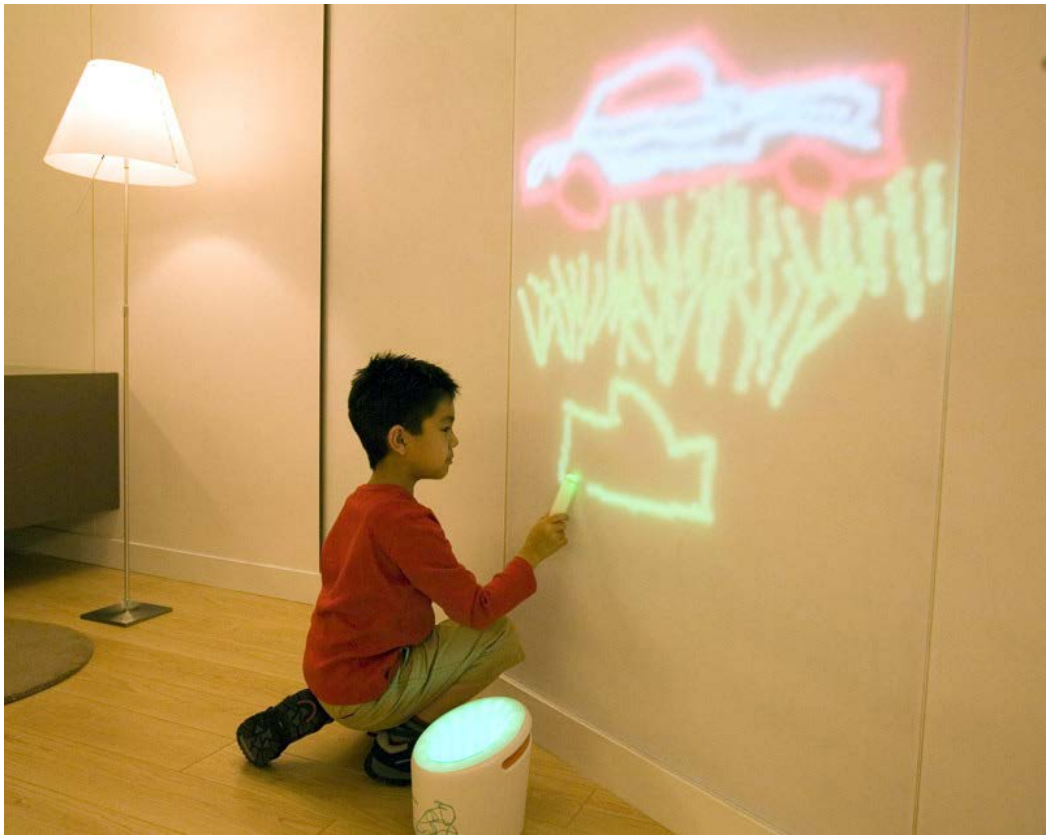
MFC Frame 구조

- 요약

Class 종류	Base Class	핵심 Function – 주 역할
Main Program Class	CWinApp	InitInstance() – Frame window를 생성 Run() – Message loop를 제공
Main Frame Class	CFrameWnd	OnCreate() – View를 생성
View Class	CWnd	OnPaint() – 화면에 출력

MFC 화면 출력

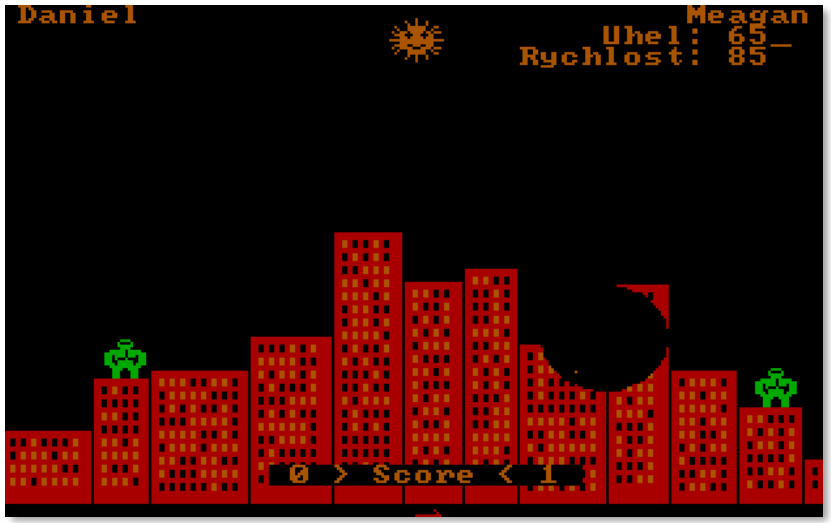
- How to draw with MFC?



MFC 화면 출력

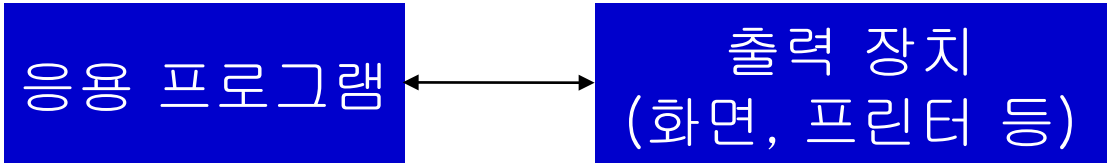
- 80년대 Windows 95 출현 이전,
  - 운영체제(SW): Disk Operating System (DOS)
  - Video card(HW): CGA(4색) → EGA(16색) → VGA(256색)
  - 게임 app: 게임회사가 video card driver를 각각 제공

→ Device-independent 개념의 시작



GDI와 DC

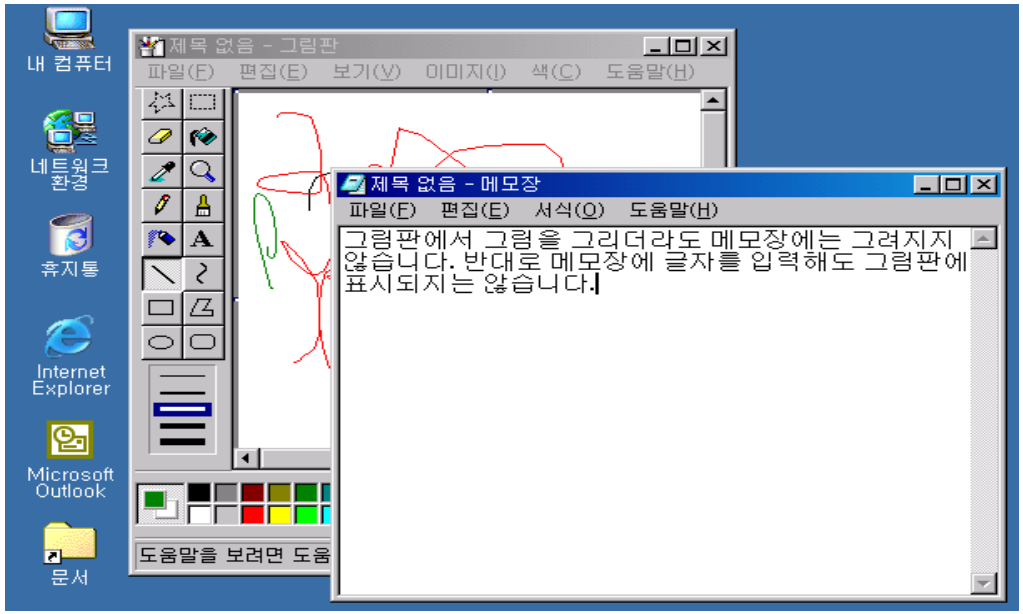
- Windows에서 출력(output)을 설계할 때 고려할 사항
  - Device-independent
    - 모니터, video card, printer 등 출력에 사용되는 주변 장치가 변경되는 경우에도 program을 수정할 필요가 없어야 함
  - Multi-tasking
    - 화면이나 기타 출력 장치를 직접 접근(direct access)하거나 독점해서 사용하는 것을 운영체제(OS) 차원에서 막아야 함



장치 의존적

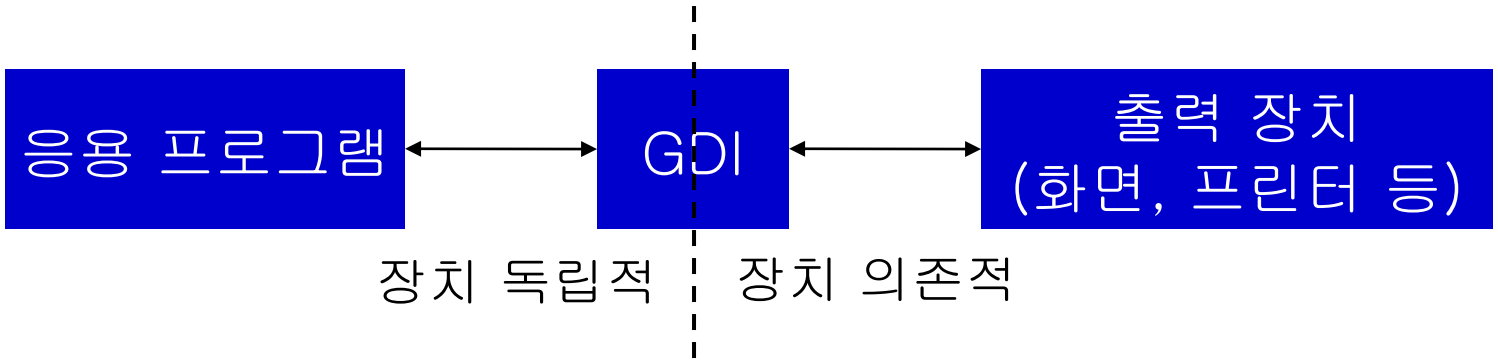
GDI와 DC

- Windows에서 화면 출력할 때 고려할 사항
  - Client 영역에 출력하려면 화면에 해당하는 Window 위치 필요
  - 화면에 여러 개의 Window가 있을 때 출력 결과가 다른 Window의 영역을 침범하면 안됨
  - 현재 출력할 화면이 다른 Window에 가려졌다면 출력을 할 수 없어야 함



GDI와 DC

- Graphics Device Interface(GDI)의 탄생
  - 운영체제의 하위 시스템 중 하나
  - 응용 program의 요청을 받아서 실제 출력 장치에 대한 출력을 담당



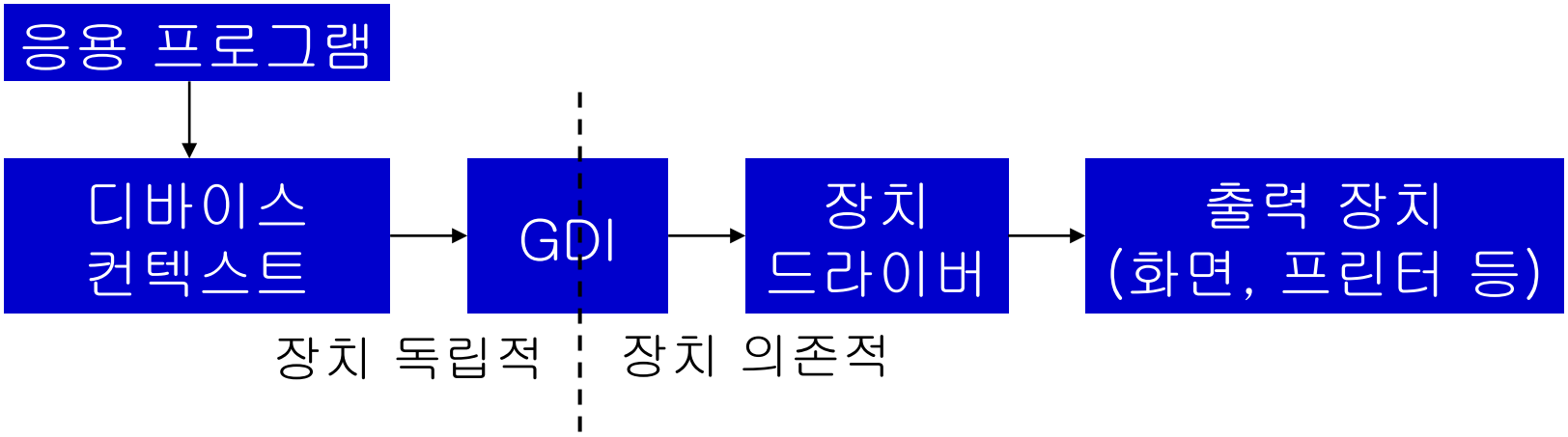
GDI와 DC

- Device Context(DC)
  - GDI가 생성하고 관리하는 data 구조체(structure)
  - Multi-tasking(multi-threading) GUI 환경에서 발생할 수 있는 여러 상황을 고려한 출력 가능

- CDC: DC의 class
- 화면에 관한 종합패키지: (화면의 상태, 그림에 대한 정보 및 그림을 그리는 함수를 제공)

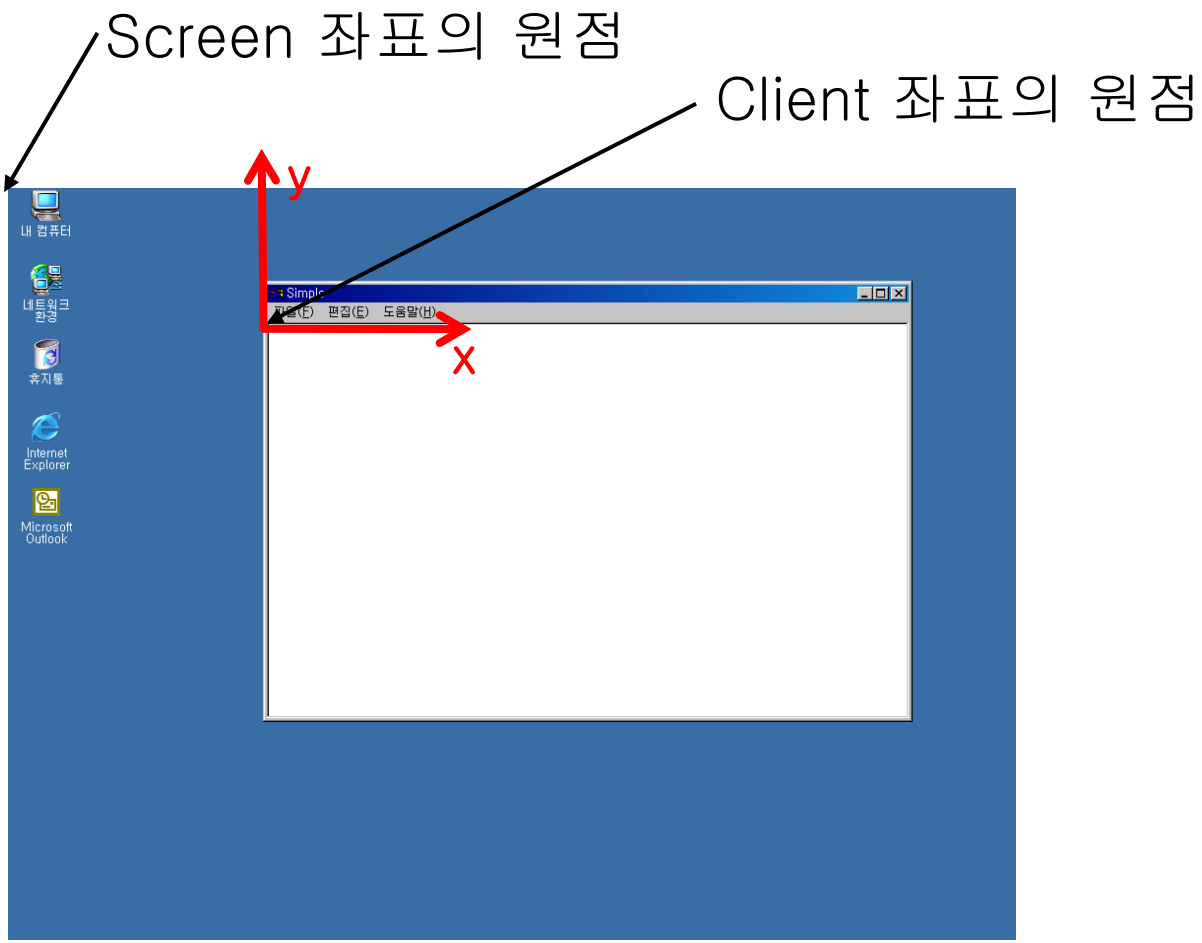
GDI와 DC

- Windows 응용 program의 출력 과정
  1. 응용 program: DC 요청
  2. 운영체제(GDI): DC 생성 및 handle(index) 제공
  3. 응용 program: DC에 그림 그리기
  4. 운영체제(GDI): 그림 그리는 상황 검토 후 출력



GDI와 DC

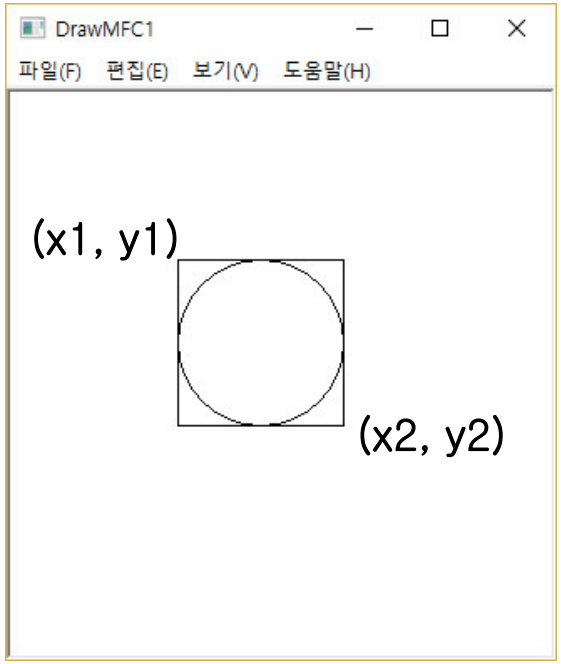
- Screen 좌표와 client 좌표



CPaintDC

- 연습: 일정한 크기의 사각형과 원을 그림 (1/2)
  - CChildView의 OnPaint() 수정

```
void CChildView::OnPaint()
{
        CPaintDC dc(this);
        dc.Rectangle(100, 100, 200, 200);
        dc.Ellipse(100, 100, 200, 200);
}
```

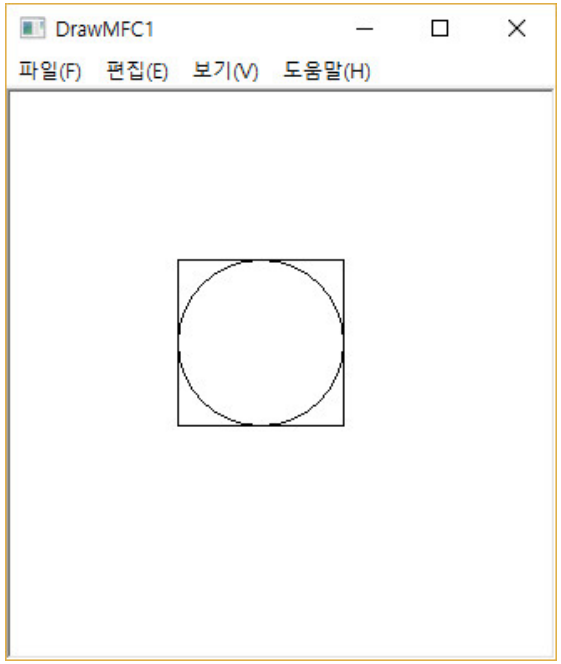


  - DC **객체 생성** (this pointer 이용)
    예) CPaintDC dc(this);
  - **객체의 멤버 함수**를 호출하여 출력
    예) dc.Rectangle(x1, y1, x2, y2);
    예) dc.Ellipse(x1, y1, x2, y2);

CPaintDC

- 연습: 일정한 크기의 사각형과 원을 그림 (2/2)
  - CChildView의 OnPaint() 수정

```
void CChildView::OnPaint()
{
        CPaintDC dc(this);
        dc.Rectangle(100, 100, 200, 200);
        dc.Ellipse(100, 100, 200, 200);
}
```

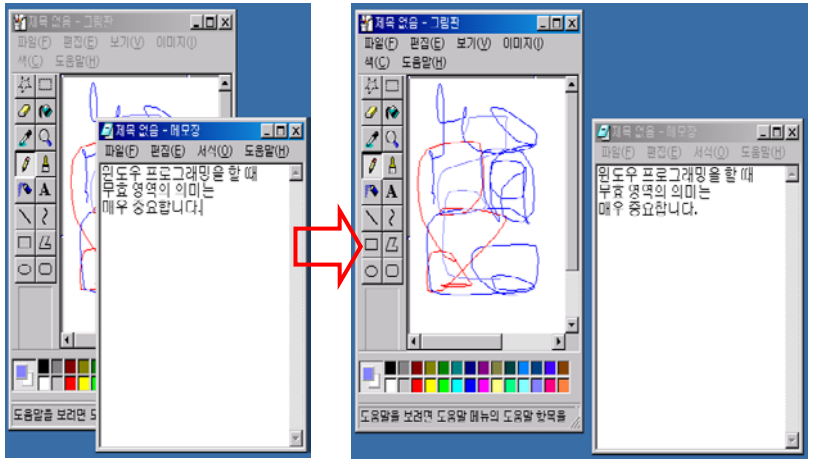


- 그림을 그릴 수 있는 위치
  - OnPaint(): WM_PAINT 발생시 자동 호출
  - 또는 다른 함수에서도 가능

- Windows Message(WM)
  - MFC에서 제공하는 기본(화면, 마우스, 키보드등과 관련된) 메세지
  - WM_PAINT: 화면을 다시 그릴 필요가 있다고 판단될 경우 발생
    → **무효 영역**(invalid region)이 생김

CPaintDC

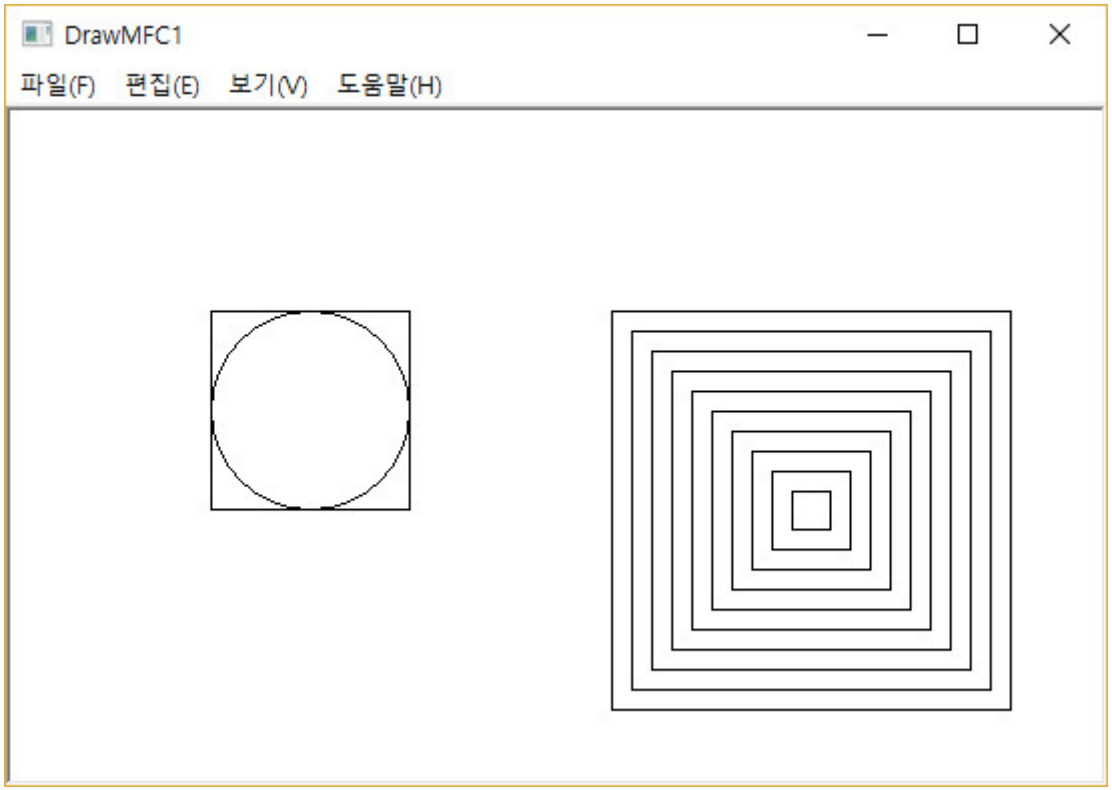
- 무효 영역: 화면을 다시 그려야 하는(WM_PAINT 발생) 상황
  - Window가 생성될 때
  - Window의 크기가 변경될 때
  - 최소화 또는 최대화 되었을 때
  - 다른 Window가 가렸다가 드러날 때
  - 또는, 필요에 따라 강제 발생도 가능



```
void CWnd::Invalidate(BOOL bErase = TRUE);
void CWnd::InvalidateRect(LPCRECT lpRect, BOOL bErase = TRUE);
```

CPaintDC

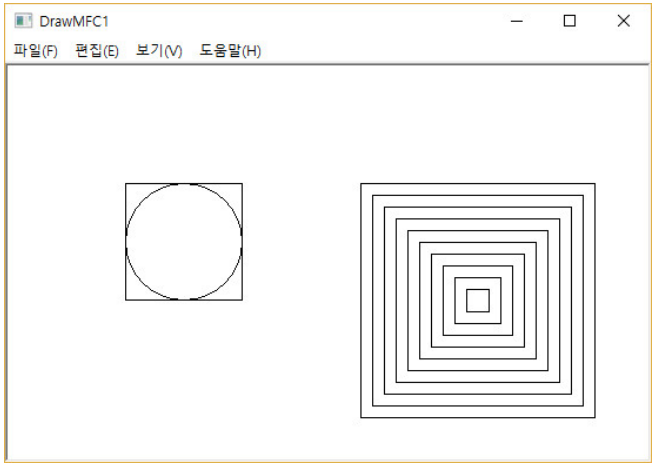
- 연습: 점점 작아지는 사각형 다수 그리기 (1/2)



CPaintDC

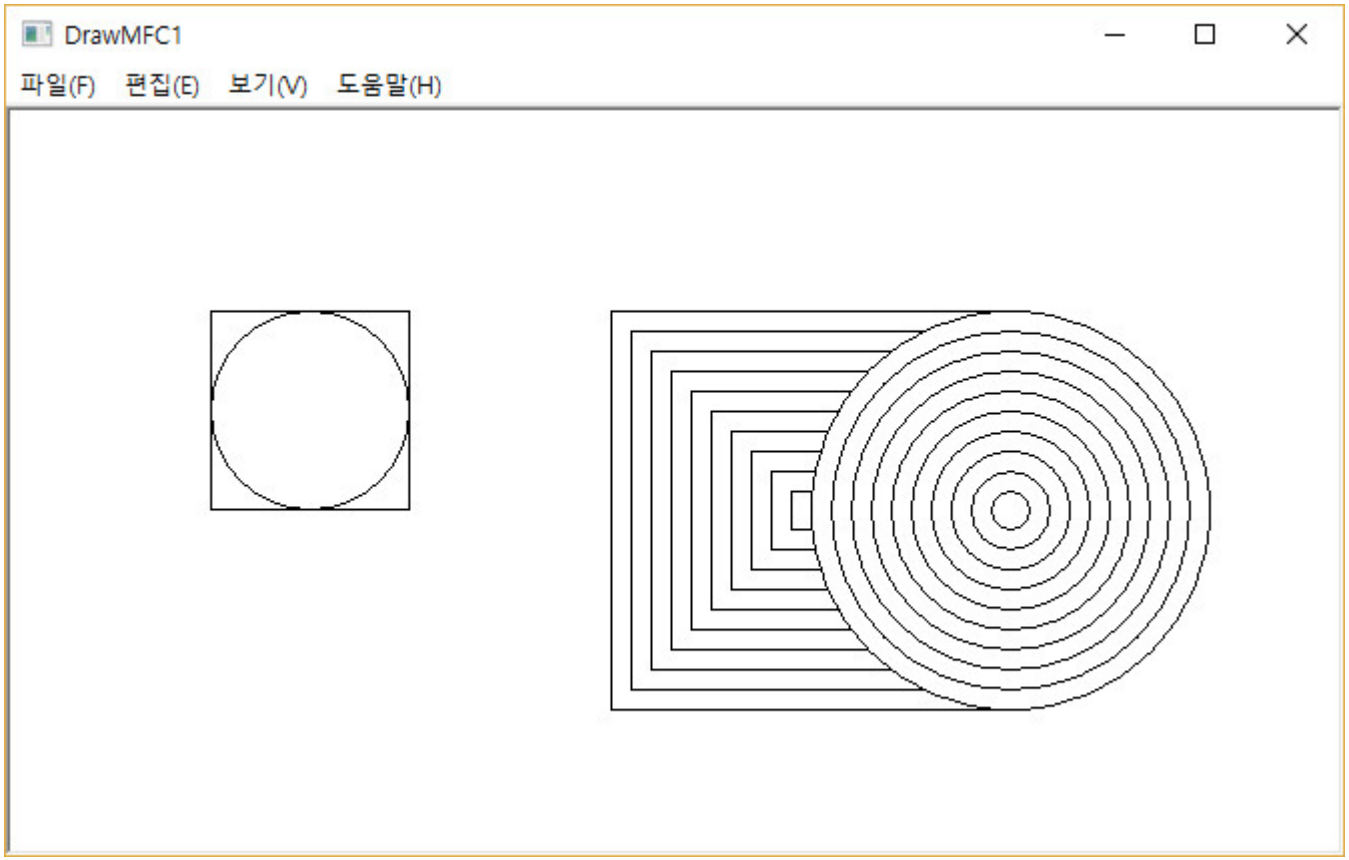
- 연습: 점점 작아지는 사각형 다수 그리기 (2/2)

```
void CMainWindow::OnPaint ()
{
    CPaintDC dc(this);

    // 점점 작아지는 사각형 그리기
    for(int i = 0; i < 10; i++)
    {
        int dd = i*10;
        dc.Rectangle(100+dd, 100+dd, 300-dd, 300-dd);
    }
}
```



CClientDC

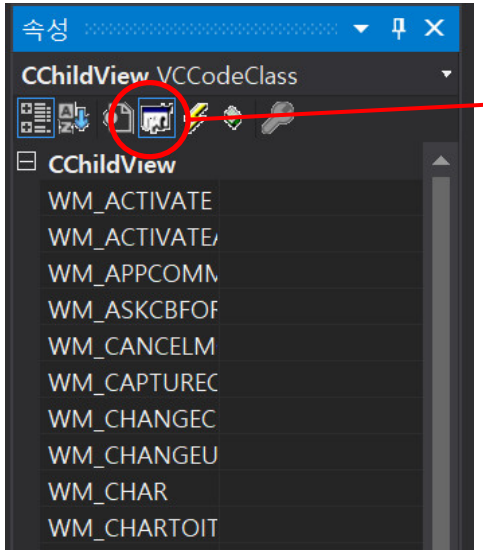
- 연습: *LMB을 누르면 점점 작아지는 원 다수 그리기 (1/4)
  *LMB: left mouse button



CClientDC

- 연습: LMB을 누르면 점점 작아지는 원 다수 그리기 (2/4)
  - Mouse button 처리기 추가: CChildView의 속성창 이용
    - 클래스 뷰 → CChildView → [*RMB] 속성

    *RMB: Right mouse button

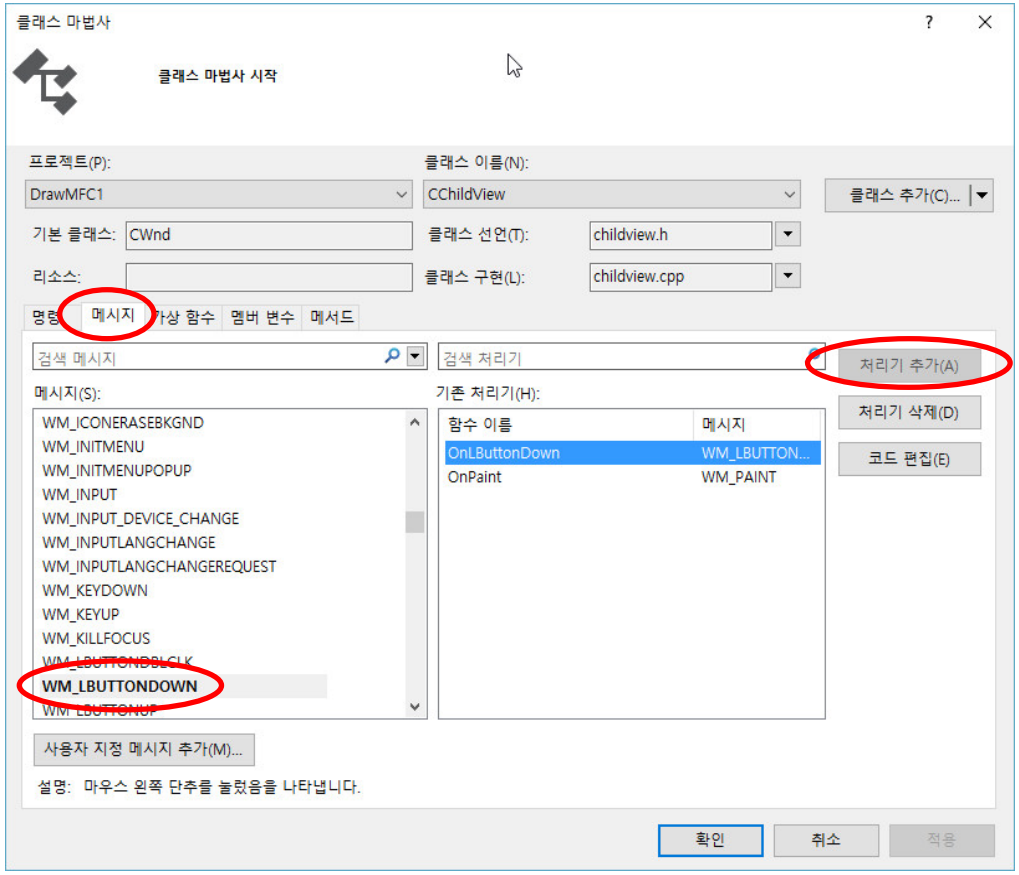


Window message 처리기 추가 버튼

Mouse button message:
- WM_LBUTTONDOWN
- WM_RBUTTONDOWN
- WM_MBUTTONDOWN
- …

CClientDC

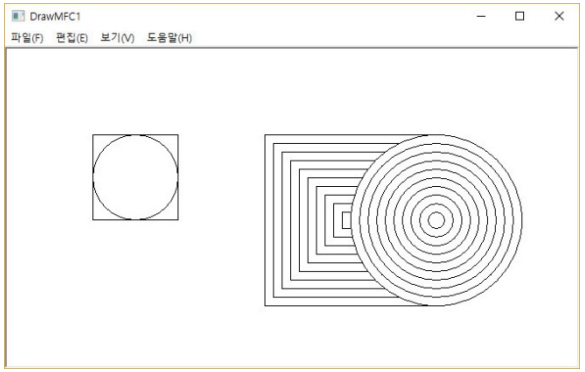
- 연습: LMB을 누르면 점점 작아지는 원 다수 그리기 (3/4)
  - Mouse button 처리기 추가: ClassWizard 이용
    - 클래스 뷰 → CChildView → [RMB] 클래스 마법사



CClientDC

- 연습: LMB을 누르면 점점 작아지는 원 다수 그리기 (4/4)
  - OnLButtonDown() 수정

```
void CChildView::OnLButtonDown(UINT nFlags, CPoint point)
{
    CClientDC dc(this);

    // 점점 작아지는 원 그리기
    for (int i = 0; i < 10; i++)
    {
        int dd = i*10;
        dc.Ellipse(400+dd, 100+dd, 600-dd, 300-dd);
    }

    CWnd::OnLButtonDown(nFlags, point);
}
```



CClientDC

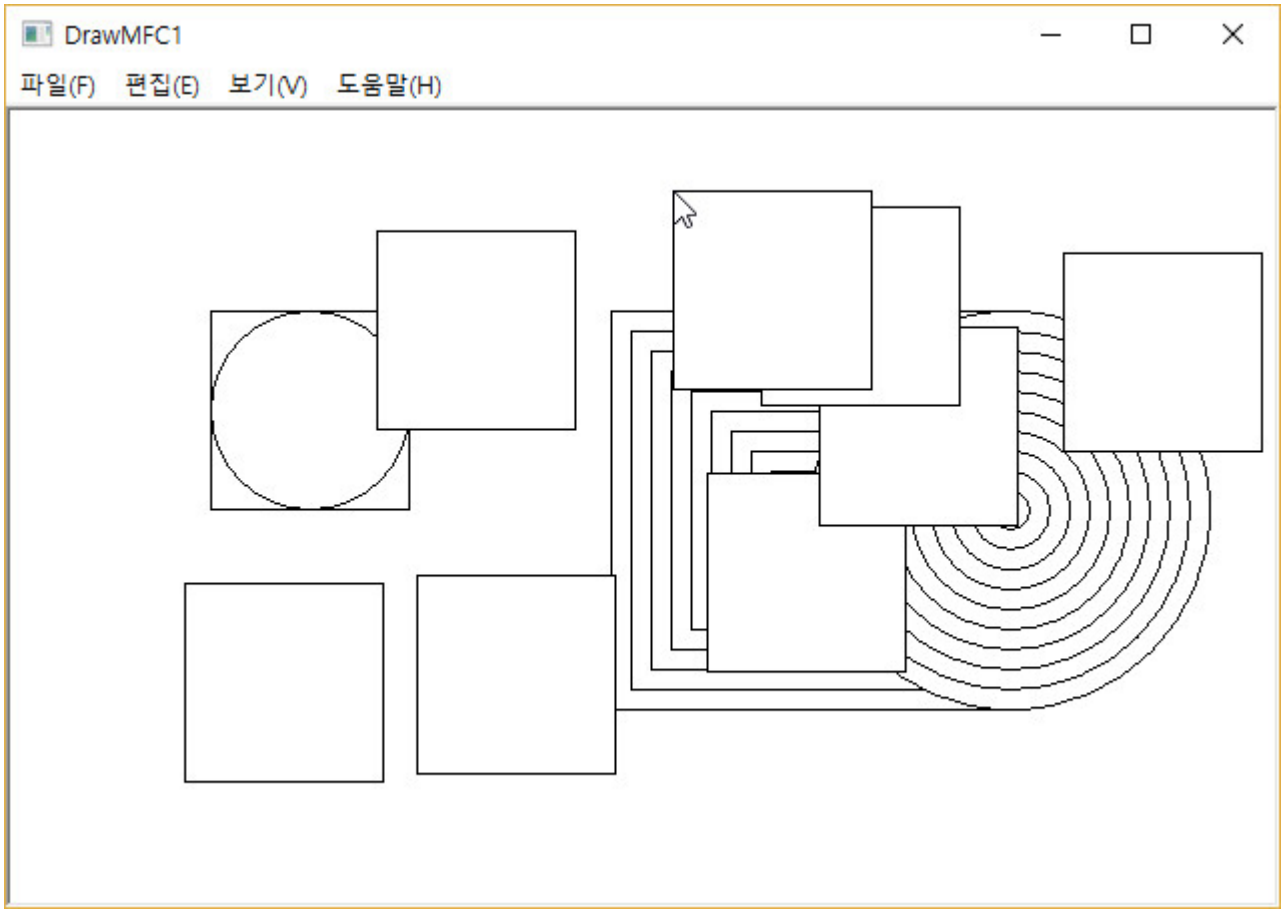
- CClientDC class
  - WM_PAINT message 처리기 이외의 부분에서 사용
  - CClientDC 사용 예

```
void CChildView::OnLButtonDown(UINT nFlags, CPoint point)
{
        CClientDC dc(this);
        // ...
}
```

- Mouse button이 눌렸을 때 그림을 그리기
  1. Mouse button이 눌리는 message 처리기 추가
  2. CClientDC 객체 생성
  3. 생성된 DC객체를 이용하여 그리기

CClientDC

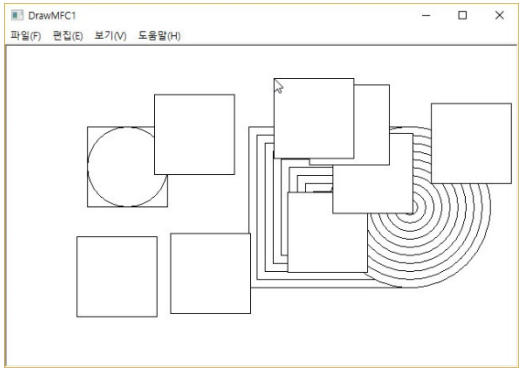
- 연습: RMB을 누른 위치에 일정한 크기의 사각형 그리기 (1/2)



CClientDC

- 연습: RMB을 누른 위치에 일정한 크기의 사각형 그리기 (1/2)
  - OnRButtonDown() message 처리기 추가 후 수정
  - 현재 mouse pointer 위치는 CPoint에서 얻어올 수 있음

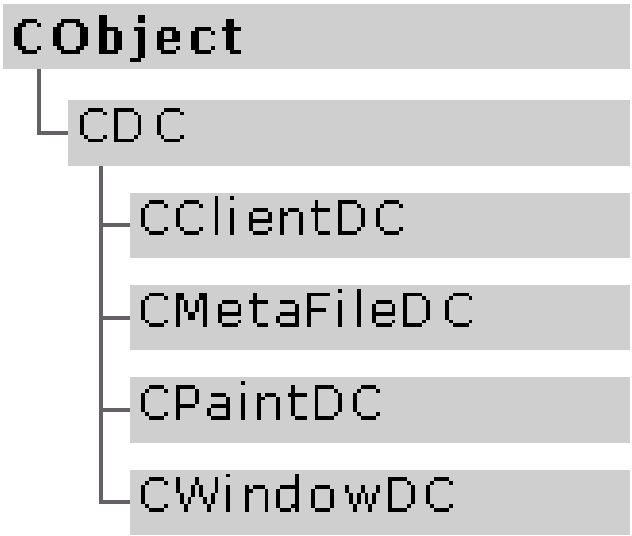
```
void CChildView::OnRButtonDown(UINT nFlags, Cpoint point)
{
    CClientDC dc(this);

    // 마우스의 위치는 point 인자(Cpoint 구조체)에서 얻어옴.
    dc.Rectangle(point.x, point.y, point.x+100, point.y+100);

    CWnd::OnLButtonDown(nFlags, point);
}
```



Windows CDC Class

- CDC class

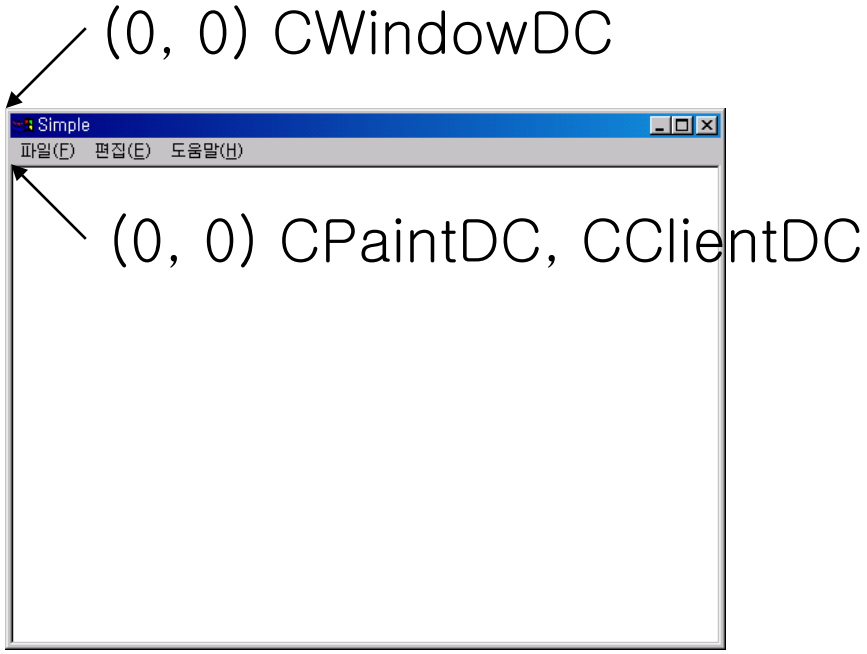
Class 이름	용도
CPaintDC	Client 영역에 출력할 때 (WM_PAINT message 처리기에서만 사용)
CClientDC	Client 영역에 출력할 때 (WM_PAINT message 처리기를 제외한 다른 모든 곳에서 사용)
CWindowDC	Window의 전체 영역(Client 영역 + 비 client 영역)에 출력할 때
CMetaFileDC	메타 파일(Metafile)에 출력할 때



Windows CDC Class

- CWindowDC 사용 방법
  - CPaintDC, CClientDC 클래스와 동일

- 원점 위치



Windows CDC Class

- 메타 파일(metafile)
  - GDI 명령어를 저장할 수 있는 파일
  - 여러 개의 저장한 GDI 명령어를 한꺼번에 수행하여 그림을 그릴 수 있음

- CMetaFileDC 사용 방법
  - CMetaFileDC 객체를 만든 후 CMetaFileDC::Create() 호출
  - Meta file 객체를 일반적인 DC 객체로 간주하고 출력 관련 member 함수를 호출
  - CMetaFileDC::Close()를 호출하면 Window 운영체제가 내부적으로 meta file을 만든 후 meta file handle(HMETAFILE 타입)을 return
  - CDC::PlayMetaFile()로 meta file을 실행

MFC vs. Win32 API

- MFC: Class의 사용으로 간단함
  - 디바이스 컨텍스트 객체 생성 (this pointer 이용)
    - CPaintDC dc(this)
  - 객체의 멤버 함수를 호출하여 출력
    - dc.TextOut(…)

- Win32 API: 더 복잡함
  - Windows 운영체제에게 device context를 요청
    - BeginPaint(hwnd, &ps);
  - 얻어낸 device context handle을 사용하여 출력
    - TextOut(hdc, 100, 100, “Hello, MFC”);
  - 운영체제에게 device context 사용이 끝났음을 알림
    - EndPaint(hwnd, &ps);

Q & A